Synthesis Comparison of Karatsuba Multiplierusing Polynomial Multiplication, Vedic Multiplier and Classical Multiplier

Sudhanshu Mishra
Department of Electronics and Telecommunication
Veer SurendraSai University of Technology, Burla
Sambalpur– 768018, Odisha, India

Manoranjan Pradhan
Department of Electronics and Telecommunication
Veer SurendraSai University of Technology, Burla
Sambalpur– 768018, Odisha, India

ABSTRACT

In this paper, the authors have compared the efficiency of the Karatsuba multiplier using polynomial multiplication with the multiplier implementing Vedic mathematics formulae (*sutras*), specifically the *Nikhilam sutra*. The multipliers have been implemented using Spartan 2 xc2s200 pq208 FPGA device having speed grade of -6. The proposed Karatsuba multiplier has been found to have better efficiency than the multipliers involving Vedic mathematics formulae.

General Terms

Karatsuba algorithm, Vedic multiplier, classical multiplication.

Keywords

Karatsuba multiplier, Vedic Mathematics, Polynomial multiplication, FPGA, Nikhilam sutra.

1. INTRODUCTION

WThe efficiency of multiplication and multipliers is a very basis for implementation in devices like the Arithmetic and Logic Units (A.L.U.), modulators, cryptosystems and many other systems involving digital signal processing. This document presents the comparison of three multiplication techniques with respect to the area requirement and speed of operation. More emphasis has been given to the Karatsuba multiplier using polynomial multiplication as it has been found to have better efficiency than the other two multipliers.

Karatsuba's multiplication algorithm uses three single digit multiplications to perform one two-digit multiplication. In their paper [1], the authors have used tensor products to express the Karatsuba algorithm in both recursive and iterative form. They have used tensor products for implementation of the recursive algorithm.

The authors Gang Zhou et al. have presented complexity analysis, in application-specific integrated circuits as well as on field-programmable gate arrays (F.P.G.A.s) and efficient implementations of bit parallel mixed Karatsuba–Ofman multipliers in [2]. By introducing the common expression sharing and the complexity analysis on odd-term polynomials, they have achieved a lower gate bound than previous ASIC discussions. They have evaluated the LUT complexity and area-time product tradeoffs on F.P.G.A.s with different computer-aided design tools. They claim that their bit parallel multipliers consume the least resources among known FPGA implementations.

Many cryptographic techniques like elliptic curve cryptography [3] and RSA algorithm [4] can be implemented very effectively using the Karatsuba algorithm.

There are Vedic mathematical techniques that can also be used for efficient multiplications in many applications. One such Vedic mathematical technique, that is, the "Nikhilam Sutra" has been compared with the proposed Karatsuba multiplier.

The authors in [4] have proposed a 16×16 multiplier using the Nikhilam Sutra and have compared its characteristics with that of another 16×16 multiplier implemented using another Vedic mathematics algorithm called the UrdhvaTiryagbhyam Sutra. They have used a carry-save adder architecture, which, as per their claim, reduces the propagation delay significantly. They have also proposed a multiplier-accumulator (MAC) unit using Vedic mathematics algorithm, the UrdhvaTiryagbhyam Sutra in [5].

2. BASICS OF KARATSUBA ALGORITHM

The basic step of Karatsuba algorithm can be used to compute the product of two large numbers a and b using three multiplications of smaller numbers, each with about half as many digits as a or b along with some additions and digit shifts.
Let a and b represent n-digit strings in some radix R. For any positive integer m less than n, the two numbers can be divided as follows:

$$a = a_i R^m + a_0.$$

$$b = b_i R^m + b_0.$$

where a_0 and b_0 are less than R^m.

The product is then

$$ab = (a_1 R^m + a_0)(b_1 R^m + b_0).$$

or, $$ab = a_1 b_1 R^{2m} + a_1 b_0 + a_0 b_1 R^m + a_0 b_0.$$

or, $$ab = u_2 R^{2m} + u_1 R^m + u_0.$$

Where,

$$u_2 = a_1 b_1,$$

$$u_1 = a_1 b_0 + a_0 b_1,$$

and $u_0 = a_0 b_0$.

These formulae require four numbers of multiplications. But, it can be observed that the value of the product *ab*can be determined using only three numbers of multiplications, at the cost of a few more number of additions in the following manner:

After obtaining,

$$u_2 = a_1 b_1 \text{and } u_0 = a_0 b_0,$$

the value of u_1 can be determined as:

$$u_1 = (a_1 + a_0)(b_1 + b_0) - u_2 - u_0.$$

since

$$u_1 = (a_1 b_0 + a_0 b_1) = (a_1 b_1 + a_1 b_0 + a_0 b_1 + a_0 b_0) - a_1 b_1 - a_0 b_0$$

or, $u_1 = (a_1 + a_0)(b_1 + b_0) - a_1 b_1 - a_0 b_0$.

2.1 Example

Let the product of numbers, 7654 and 6789, be determined using Karatsuba algorithm. For calculating the product of 7654 and 6789, the values of *R* and *m* can be chosen as 10 and 2 respectively.

$R = 10 \text{and } m = 2$

$$2178 = 21 \times 10^2 + 78$$

$$5423 = 54 \times 10^2 + 23$$

$$u_2 = 21 \times 54 = 1134$$

$$u_0 = 78 \times 23 = 1794$$

$$u_1 = (21 + 78)(54 + 23) - u_2 - u_0$$

or, $u_1 = (99 \times 77) - 1134 - 1794$

or, $u_1 = 7623 - 1134 - 1794 = 4695$

Therefore, the product of 2178and 5423 can be calculated as:

$$2178 \times 5423 = (1134 \times 10000) + (4695 \times 100) + 1794$$

or, $2178 \times 5423 = 11340000 + 469500 + 1794 = 11811294$

3. GENERAL METHOD OF POLYNOMIAL MULTIPLICATION

Usually multiplication of polynomials is done in the following manner:

Let there be two degree-d polynomials with n = d+ 1 coefficients:

$$A(x) = \sum_{i=0}^{d} a_i x^i$$

and

$$B(x) = \sum_{i=0}^{d} b_i x^i$$

Then the product of A(x) and B(x) can be written as

$$C(x) = A(x)B(x) = \sum_{i=0}^{d} \sum_{j=0}^{d} a_i \, b_j$$

The polynomial C(x) can be obtained with n2 multiplications and (n − 1)2 additions.

4. KARATSUBA ALGORITHM

This section describes the general technique for multiplication of two polynomials of any arbitrary degree with n number of coefficients using the Karatsuba algorithm:

Let there be two degree-d polynomials with n number of coefficients such that n = d + 1 given by:

$$A(x) = \textstyle\sum_{i=0}^{d} a_i \, x^i \qquad (1)$$

and

$$B(x) = \textstyle\sum_{i=0}^{d} b_i \, x^i \qquad (2)$$

Then a set of auxiliary variables can be defined as:

$$D_i = a_i b_i, \forall \; i = 0, 1, 2, \ldots, n-1 \qquad (3)$$

and

$$D_{p,q} = (a_p + a_q)(b_p + b_q), \forall \; i \; such \; that \; p + q = i \; and \; q > p \geq 0 \qquad (4)$$

The product of *A(x)* and *B(x)*, that is, *C(x)* can be given as:

$$A(x) \; B(x) = \textstyle\sum_{i=0}^{2n-2} c_i \; x^i \, . \qquad (5)$$

Where the values of c_i can be given as:

$$c_0 = D_0,$$

$$c_{2n-2} = D_{n-1},$$

and

$$c_i = \begin{cases} \sum_{p+q=i, q>p\geq 0} D_{p,q} - \sum_{p+q=i, q>p\geq 0} (D_p + D_q), \\ for \; odd \; values \; of \; i, 0 < i < 2n-2 \\ \sum_{p+q=i, q>p\geq 0} D_{p,q} - \sum_{p+q=i, q>p\geq 0} (D_p + D_q) + D_{\frac{i}{2}}, \\ for \; even \; values \; of \; i, 0 < i < 2n-2 \end{cases}$$

Therefore for multiplying one-degree polynomials, that is, d = 1, n = d +1=2, using the equations (1), (2) and (3),

$$A(x) = a_1 x + a_0 \qquad \text{[from eq. (1)]}$$

$$B(x) = b_1 x + b_0 \qquad \text{[from eq. (2)]}$$

Then the product, *C(x) = A(x) B(x)* can be determined in the following manner:

The auxiliary variables are:

$$D_0 = a_0 b_0$$

$$D_1 = a_1 b_1$$

and $D_{0,1} = (a_0 + a_1)(b_0 + b_1)$

Now, $c_0 = D_0 = a_0 b_0$

$$c_{2n-2} = c_2 = D_1 = a_1 b_1$$

and

$$c_1 = D_{0,1} - (D_0 + D_1) = (a_0 + a_1)(b_0 + b_1) - (a_0 b_0 + a_1 b_1)$$

$$= (a_0 b_1 + a_1 b_0)$$

Therefore, the polynomial (product) *C(x)* can be written as:

$$C(x) = D_1x^2 + (D_{0,1} - (D_0 + D_1))\,x + D_0 \quad (6)$$

The above equation when expanded becomes:

$$C(x) = (a_1b_1)x^2 + (a_0b_1 + a_1b_0)\,x + a_0b_0$$

which is the product of *A(x)* and *B(x)*.

Similarly, for multiplication of two - degree polynomials that is, d = 1, n = d +1=2, using the equations (1), (2) and (3),

$$A(x) = a_2x^2 + a_1x + a_0 \qquad \text{[from eq. (1)]}$$

$$B(x) = b_2x^2 + b_1x + b \qquad \text{[from eq. (2)]}$$

The auxiliary variables are:

$$D_0 = a_0b_0$$

$$D_1 = a_1b_1$$

$$D_2 = a_2b_2$$

$$D_{0,1} = (a_0 + a_1)(b_0 + b_1)$$

$$D_{0,2} = (a_0 + a_2)(b_0 + b_2)$$

and $$D_{1,2} = (a_1 + a_2)(b_1 + b_2)$$

Now, $$c_0 = D_0 = a_0b_0$$

$$c_{2n-2} = c_4 = D_1 = a_1b_1$$

$$c_1 = D_{0,1} - (D_1 + D_0) = (a_0b_1 + a_1b_0)$$

$$c_2 = D_{0,2} - (D_2 + D_0) + D_1 = (a_0b_2 + a_2b_0 + a_1b_1)$$

and $$c_2 = D_{1,2} - (D_1 + D_2) = (a_1b_2 + a_2b_1)$$

Hence, the polynomial (product) *C(x)* can be written as:

$$C(x) = D_2x^4 + \left(D_{1,2} - (D_1 + D_2)\right)x^3 + \left(D_{0,2} - (D_2 + D_0) + D_1\right)x^2 + (D_{0,1} - (D_1 + D_0))\,x + D_0 \quad (7)$$

The above equation can be expanded as:

$$\begin{aligned} C(x) = (a_2b_2)x^4 &+ (a_1b_2 + a_2b_1)x^3 \\ &+ (a_0b_2 + a_2b_0 + a_1b_1)x^2 \\ &+ (a_0b_1 + a_1b_0)x + a_0b_0 \end{aligned}$$

which is the product of the polynomials *A(x)* and *B(x)*.

In their paper [6], Weimerskirch and Paar have presented a detailed analysis of the Karatsuba algorithm using recursive as well as iterative approach.

5. VEDIC MULTIPLIER (NIKHILAM SUTRA)

The literal meaning of Nikhilam Sutra is “all from 9 and last from 10”. This algorithm is more efficient for multiplication of large numbers. It finds out the complement of the large number from its nearest base to perform the multiplication. The Nikhilam Sutra is explained by considering the multiplication of two decimal numbers (8 × 9) where the base is 10 which is nearest to as well as greater than these two numbers.

Table 1: Multiplication of two decimal numbers (8 × 9) using ‘Nikhilam’ Sutra

Column 1	*Column 2*
8	10 - 8 = 2
9	10 - 9 =1
(8-1) or (9-2) = 7	2×1 = 2

Table 1 shows the multiplication of two one digit decimal numbers using the Nikhilam sutra. The first two rows of column 1 show the multiplier and multiplicand. The first two rows of the second column display the complements of the multiplier and multiplicand (i.e. base 10). The third row of first column represents the left-hand side (L.H.S.) of the product and that of the second column represents the right-hand side (R.H.S.) of the product. The R.H.S. of the product can be obtained by multiplying the numbers of the Column 2 (2 × 1= 2). However the surplus portion on the R.H.S. is carried over to Left. The left hand side (L.H.S.) of the product can be found by cross subtracting the second number of Column 2 from the first number of Column 1 or vice versa, i.e., 8 – 1 = 7 or 9 – 2 = 7. The final result is obtained by concatenating the digits in R.H.S. and L.H.S. (Answer = 72).

6. SYNTHESIS RESULTS AND COMPARISONS

The multipliers, namely, Karatsuba multiplier, Vedic multiplier and a classical multiplier were implemented using Spartan 2s200pq208 FPGA device having a speed grade of -6. The codes were written in VHDL and they were simulated and synthesised using Xilinx ISE 10.1 simulator. The observations have been tabulated in Table 2.

Table 2 shows the statistics of device usage and combinational path delay for 8×8 Karatsuba multiplier, classical Multiplier and Vedic multiplier. For Karatsuba multiplier the number of slices is the least, i.e. 26 as compared to that of the classical multiplier and the Vedic multiplier which are 38 and 62 respectively. Also, the number of four input LUTs and number of bonded IOBs are less for the proposed Karatsuba multiplier than the other two multipliers. These observations prove that area requirement for the Karatsuba multiplier is least. The maximum combinational path delay for the Karatsuba multiplier is least as well, that is 12.338ns as compared to 15.656ns and 27.340ns for the classical multiplier and the Vedic multiplier respectively. This proves that the time delay is also least for the Karatsuba multiplier as compared to the other two multipliers.

Table 2.Comparison of device utilization and combinational path delay of 8×8 Karatsuba multiplier, classical multiplier, and Vedic multiplier

device (Spartan 2 xc2s200 pq208)	***number of slices***	***number of 4 input LUTs***	***number of bonded IOBs***	***maximum combinational path delay***
8×8 (Karatsuba Multiplier)	26 out of 2352	45 out of 4704 (0%)	31 out of 140 (22%)	12.338ns

	(1%)			
8×8 (Classical Multiplier)	38 out of 2352 (1%)	73 out of 4704 (1%)	32 out of 140 (22%)	15.656ns
8×8 (Vedic Multiplier)	62 out of 2352 (2%)	113 out of 4704 (2%)	32 out of 140 (22%)	27.340ns

The figure 1 exhibits the histogram representation of the performance of the three multipliers with respect to device utilization and time requirements (path delay). It can be remarked from the figure that the proposed Karatsuba multiplier requires less space for its implementation and simultaneously it requires less time for its execution.

Thus, from figure 1 and table 2, it can be concluded that the proposed Karatsuba multiplier is more efficient in terms of both space and time requirements than the Vedic multiplier (using *Nikhilam Sutra)* and the classical multiplier.

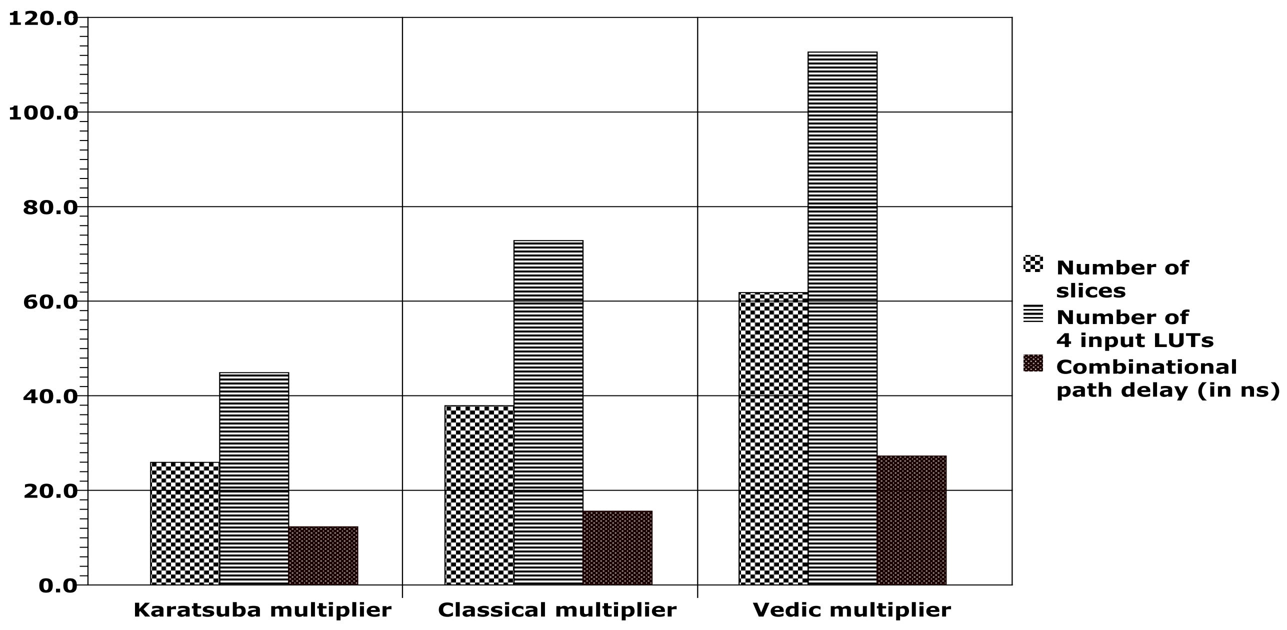

Figure 1: Histogram showing the comparison of device utilization and combinational path delay of different multipliers

7. CONCLUSION
The combinational path delay and device utilization of 8×8 Karatsuba multiplier, Vedic multiplier and a classical multiplier has been compared. The proposed Karatsuba multiplier shows speed improvement as compared to Vedic multiplier and the classical multiplier. This may be useful for applications involving high speed multiplication.

8. REFERENCES
[1] Chin-Bou Liu, Chua-Huang Huang, and Chin-Laung. Lei, "Design and Implementation of long-digit Karatsuba's multiplication algorithm using tensor product formulation", in Ninth workshop on compiler techniques for high performance computing, 2003, pp. 1-8.

[2] G. Zhou, H. Michalik, and L. Hinsenkamp, "Complexity Analysis and Efficient Implementations of Bit Parallel Finite Field Multipliers Based on Karatsuba-Ofman Algorithm on FPGAs", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 18, No.7, 2010, pp.1057-1066.

[3] Z. Dyka and P. Langendoerfer, "Area Efficient Hardware Implementation of Elliptic Curve Cryptography by Iteratively Applying Karatsuba's Method", in Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (IEEE Computer Society), 2005,1530-1591/05.

[4] M. Pradhan , R. Panda and S.K. Sahu, "Speed Comparison of 16×16 Vedic Multipliers", International Journal of Computer Applications, Vol. 21, No. 6, 2011, pp. 16-19.

[5] M. Pradhan , R. Panda and S.K. Sahu, "MAC Implementation using Vedic Multiplication Algorithm", International Journal of Computer Applications, Vol. 21, No. 7, 2011, pp. 26-28.

[6] A. Weimerskirch and C. Paar, "Generalizations of the Karatsuba algorithm for efficient implementations," 2006. [Online]. Available: http:// eprint.iacr.org/2006/224.pdf

[7] C. Rebeiro and D. Mukhopadhyay, "Power attack resistant efficient FPGA architecture for Karatsuba multiplier," in *21st International Conference on VLSI Design*, 2008, pp. 706–711.

[8] M. Markovic, T. Unkasevic, and G. Dordevic, "RSA Algorithm Optimization On Assembler Of Ti

Tms320c54x Signal Processors", in proceedings of European Association for Signal Processing, 2002. Available at: http://www.eurasip.org/Proceedings/Eusipco/2002/articles/paper189.pdf.

[9] S.R. Vaidya and D.R. Dandekar, "Performance Comparison of Multipliers for Power-Speed Trade-off in VLSI Design", in 12th International Conference on Networking, VLSI and Signal Processing ,2010, 262-266.

[10] Leonard Gibson Moses S and Thilagar M, "VLSI Implementation of High Speed DSP algorithms using Vedic Mathematics", International Journal of Computer Communication and Information System, 2010,Vol. 2, No. 1, pp. 119-122.

[11] Duif, N. 2011 Smart card implementation of a digital signature scheme for Twisted Edwards curves. Master thesis. Student number: 0554878. Department of Mathematics and Computer Science. TechnischeUniversiteit Eindhoven.

[12] BogdanPasca. 2011. High-performance floating-point computing on reconfigurable circuits. Doctoral thesis. Superior Normal School Of Lyon (ÉcoleNormaleSupérieure De Lyon). Laboratory of Parallel Computing (Laboratoire de l'Informatique du Parallélisme). Graduate School of Mathematics and Computer Science from Lyon.